main.py

Save Run

```python
def intToRoman(num):
    values = [1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1]
    symbols = ["M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV",
        "I"]

    roman_numeral = ''
    for i in range(len(values)):
        while num >= values[i]:
            roman_numeral += symbols[i]
            num -= values[i]
    return roman_numeral
print(intToRoman(3))
print(intToRoman(58))
print(intToRoman(1994))

```

Output Clear

```
III
LVIII
MCMXCIV

=== Code Execution Successful ===
```

main.py

Save Run

```python
def isValid(s):
    stack = []
    mapping = {')': '(', '}': '{', ']': '['}
    for char in s:
        if char in mapping.values():
            stack.append(char)
        elif char in mapping:
            if not stack or stack.pop() != mapping[char]:
                return False
        else:
            return False
    return not stack
print(isValid("()"))
print(isValid("()[]{}"))
print(isValid("(]"))
print(isValid("([)]"))
print(isValid("{[]}"))

```

Output Clear

```
1 -> 2 -> 3 -> 5 ->
=== Code Execution Successful ===
```

main.py

Save Run

```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
def removeNthFromEnd(head, n):
    dummy = ListNode(0)
    dummy.next = head
    first = dummy
    second = dummy
    for _ in range(n + 1):
        first = first.next
    while first is not None:
        first = first.next
        second = second.next
    second.next = second.next.next
    return dummy.next
head = ListNode(1)
head.next = ListNode(2)
head.next.next = ListNode(3)
head.next.next.next = ListNode(4)
head.next.next.next.next = ListNode(5)
head = removeNthFromEnd(head, 2)
current = head
while current:
    print(current.val, end=" -> ")
    current = current.next
```

Output

Clear

```
1 -> 2 -> 3 -> 5 ->
=== Code Execution Successful ===
```

main.py

Save Run

```
    res = []
    nums.sort()
    n = len(nums)
    for i in range(n):
        for j in range(i + 1, n):
            left, right = j + 1, n - 1
            while left < right:
                total = nums[i] + nums[j] + nums[left] + nums[right]
                if total == target:
                    res.append([nums[i], nums[j], nums[left], nums[right]])
                    while left < right and nums[left] == nums[left + 1]:
                        left += 1
                    while left < right and nums[right] == nums[right - 1]:
                        right -= 1
                    left += 1
                    right -= 1
                elif total < target:
                    left += 1
                else:
                    right -= 1
            while j + 1 < n and nums[j] == nums[j + 1]:
                j += 1
        while i + 1 < n and nums[i] == nums[i + 1]:
            i += 1
    return res
print(fourSum([1, 0, -1, 0, -2, 2], 0))
```

Output Clear

```
[[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]

=== Code Execution Successful ===
```

main.py

Save Run

```python
def max_area(height):
    max_area = 0
    left = 0
    right = len(height) - 1

    while left < right:
        width = right - left
        h = min(height[left], height[right])
        max_area = max(max_area, width * h)

        if height[left] < height[right]:
            left += 1
        else:
            right -= 1

    return max_area
```

Output

Clear

```
Input: height = [1,8,6,2,5,4,8,3,7]
Output: 49

=== Code Execution Successful ===
```

main.py

Save Run

```python
def three_sum(nums):
    nums.sort()
    res = []
    for i in range(len(nums)-2):
        if i > 0 and nums[i] == nums[i-1]:
            continue
        l, r = i+1, len(nums)-1
        while l < r:
            total = nums[i] + nums[l] + nums[r]
            if total < 0:
                l += 1
            elif total > 0:
                r -= 1
            else:
                res.append([nums[i], nums[l], nums[r]])
                while l < r and nums[l] == nums[l+1]:
                    l += 1
                while l < r and nums[r] == nums[r-1]:
                    r -= 1
                l += 1
                r -= 1
    return res
nums = [-1, 0, 1, 2, -1, -4]
print(three_sum(nums))
```

Output Clear

```
[[-1, -1, 2], [-1, 0, 1]]

=== Code Execution Successful ===
```

main.py

Save Run

```python
def threeSumClosest(nums, target):
    nums.sort()
    closest_sum = float('inf')

    for i in range(len(nums) - 2):
        left, right = i + 1, len(nums) - 1

        while left < right:
            current_sum = nums[i] + nums[left] + nums[right]

            if abs(target - current_sum) < abs(target - closest_sum):
                closest_sum = current_sum

            if current_sum < target:
                left += 1
            else:
                right -= 1
                return closest_sum
nums = [-1, 2, 1, -4]
target = 1
result = threeSumClosest(nums, target)
print(result)

```

Output Clear

```
2

=== Code Execution Successful ===
```

main.py

Save Run

```python
def romanToInt(s):
    roman_values = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M':
        1000}
    result = 0
    prev_value = 0

    for char in s:
        value = roman_values[char]
        if value > prev_value:
            result += value - 2 * prev_value
        else:
            result += value
        prev_value = value

    return result
print(romanToInt("III"))
print(romanToInt("LVIII"))
print(romanToInt("MCMXCIV"))

```

Output Clear

```
3
58
1994

=== Code Execution Successful ===
```

main.py

Save Run

```python
def letterCombinations(digits):
    if not digits:
        return []
    mapping = {
        '2': 'abc',
        '3': 'def',
        '4': 'ghi',
        '5': 'jkl',
        '6': 'mno',
        '7': 'pqrs',
        '8': 'tuv',
        '9': 'wxyz'
    }
    def backtrack(index, path):
        if len(path) == len(digits):
            combinations.append(''.join(path))
            return
```

Output Clear

```
=== Code Execution Successful ===
```

main.py

Save Run

```python
def longestCommonPrefix(strs):
    if not strs:
        return ""

    strs.sort()
    prefix = ""

    for i in range(len(strs[0])):
        if strs[0][i] == strs[-1][i]:
            prefix += strs[0][i]
        else:
            break

    return prefix

print(longestCommonPrefix(["flower", "flow", "flight"]))
print(longestCommonPrefix(["dog", "racecar", "car"]))
```

Output Clear

```
fl


=== Code Execution Successful ===
```